

DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG

```
DDDDDDDD  BBBB BBBB  GGGGGGGG  NN  NN  EEEEEEEEE  RRRRRRRR  MM  MM  SSSSSSSS  GGGGGGGG
DDDDDDDD  BBBB BBBB  GGGGGGGG  NN  NN  EEEEEEEEE  RRRRRRRR  MM  MM  SSSSSSSS  GGGGGGGG
DD  DD  BB  BB  GG  NN  NN  EE  RR  RR  MMMM  MMMM  SS  GG
DD  DD  BB  BB  GG  NN  NN  EE  RR  RR  MMMM  MMMM  SS  GG
DD  DD  BB  BB  GG  NNNN  NN  EE  RR  RR  MM  MM  SS  GG
DD  DD  BB  BB  GG  NNNN  NN  EE  RR  RR  MM  MM  SS  GG
DD  DD  BBBB BBBB  GG  NN  NN  EEEEEEE  RRRRRRRR  MM  MM  SSSSSS  GG
DD  DD  BBBB BBBB  GG  NN  NN  EEEEEEE  RRRRRRRR  MM  MM  SSSSSS  GG
DD  DD  BB  BB  GG  GG  NN  NN  EE  RR  RR  MM  MM  SS  GG
DD  DD  BB  BB  GG  GG  NN  NN  EE  RR  RR  MM  MM  SS  GG
DD  DD  BB  BB  GG  GG  NN  NN  EE  RR  RR  MM  MM  SS  GG
DD  DD  BB  BB  GG  GG  NN  NN  EE  RR  RR  MM  MM  SS  GG
DDDDDDDD  BBBB BBBB  GGGGGG  NN  NN  EEEEEEEEE  RR  RR  MM  MM  SSSSSSSS  GGGGGG
DDDDDDDD  BBBB BBBB  GGGGGG  NN  NN  EEEEEEEEE  RR  RR  MM  MM  SSSSSSSS  GGGGGG
```

```
LL  IIIII  SSSSSSSS
LL  IIIII  SSSSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SSSSSS
LL  II  SSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SS
LLLLLLLLLL  IIIII  SSSSSSSS
LLLLLLLLLL  IIIII  SSSSSSSS
```



```
1 0001 0 MODULE DBGNERMSG (IDENT = 'V04-000') =
2 0002 1 BEGIN
3 0003 1
4 0004 1
5 0005 1
6 0006 1
7 0007 1
8 0008 1
9 0009 1
10 0010 1
11 0011 1
12 0012 1
13 0013 1
14 0014 1
15 0015 1
16 0016 1
17 0017 1
18 0018 1
19 0019 1
20 0020 1
21 0021 1
22 0022 1
23 0023 1
24 0024 1
25 0025 1
26 0026 1
27 0027 1
28 0028 1
29 0029 1
30 0030 1
31 0031 1
32 0032 1
33 0033 1
34 0034 1
35 0035 1
36 0036 1
37 0037 1
38 0038 1
39 0039 1
40 0040 1
41 0041 1
42 0042 1
43 0043 1
44 0044 1
45 0045 1
46 0046 1
47 0047 1
48 0048 1
49 0049 1
50 0050 1
51 0051 1
52 0052 1
53 0053 1
54 0054 1
55 0055 1
56 0056 1
57 0057 1

0001 0 MODULE DBGNERMSG (IDENT = 'V04-000') =
0002 1 BEGIN
0003 1
0004 1
0005 1
0006 1
0007 1
0008 1
0009 1
0010 1
0011 1
0012 1
0013 1
0014 1
0015 1
0016 1
0017 1
0018 1
0019 1
0020 1
0021 1
0022 1
0023 1
0024 1
0025 1
0026 1
0027 1
0028 1
0029 1
0030 1
0031 1
0032 1
0033 1
0034 1
0035 1
0036 1
0037 1
0038 1
0039 1
0040 1
0041 1
0042 1
0043 1
0044 1
0045 1
0046 1
0047 1
0048 1
0049 1
0050 1
0051 1
0052 1
0053 1
0054 1
0055 1
0056 1
0057 1

*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****

++
FACILITY:      DEBUG

ABSTRACT:

Version 3 debugger error output routines are contained in this module. In
contrast to the version 2 debugger, error messages are not handled by the
exception handling mechanism. That is, error messages are not SIGNALed. The
routines in this module call SYSS$PUTMSG to recover and format the DEBUG
messages from the system message file. The address of the version 2 debugger
routine dbg$out_message is supplied as an action routine. It is this routine
which actually outputs the message. SYSS$PUTMSG is used instead of SYSS$GETMSG
because the parameters to SYSS$PUTMSG resemble the the vector of longwords
formed by a SIGNAL, a format which dbg$out_message expects.

ENVIRONMENT:   VAX/VMS

AUTHOR:        David Plummer, CREATION DATE:   4/10/80

MODIFIED BY:   David Plummer, 10-Jul-80, DLP

2.2-001        10-Jul-80        DLP        Added check for a null message vector ptr

R. Title       Feb 1983        Added parse and execute of DUMP
command to this module (for lack
of a better place to put it).
```

DBGNERMSG
V04-000

I 6
16-Sep-1984 01:42:49 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:17:11 [DEBUG.SRC]DBGNERMSG.B32;1

Page 2
(1)

:	58	0058	1	:	
:	59	0059	1	:	
:	60	0060	1	:	
:	61	0061	1	:	VERSION:
:	62	0062	1	:	
:	63	0063	1	:	--

V02.2-002

This command is used by developers
to dump DEBUG internals.


```

65 0064 1 |
66 0065 1 | TABLE OF CONTENTS:
67 0066 1 |
68 0067 1 |
69 0068 1 | FORWARD ROUTINE
70 0069 1 |     DBG$NPARSE_DUMP,           | Parse DUMP command
71 0070 1 |     DBG$NEXECUTE_DUMP,        | Execute DUMP command
72 0071 1 |     DBG$NOUT_INFO,           | Outputs an informational message
73 0072 1 |     DBG$NMAKE_ARG_VECT,       | Constructs a standard message argument vector
74 0073 1 |     DBG$NOUT_ARG_VECT : NOVALUE, | Outputs a message argument vector
75 0074 1 |     DBG$NSYNTAX_ERROR;        | Constructs a syntax error message vect
76 0075 1 |
77 0076 1 |
78 0077 1 | REQUIRE FILES:
79 0078 1 |
80 0079 1 |
81 0080 1 | REQUIRE 'SRC$:DBGPROLOG.REQ';
82 0214 1 |
83 0215 1 |
84 0216 1 | EXTERNAL REFERENCES:
85 0217 1 |
86 0218 1 | EXTERNAL ROUTINE
87 0219 1 |     DBG$ANALYZE_HASH: NOVALUE, | Dump info about hash chains
88 0220 1 |     DBG$DUMP_GLOBAL: NOVALUE,  | Dump info about GST
89 0221 1 |     DBG$DUMP_SAT: NOVALUE,     | Dump info about SAT
90 0222 1 |     DBG$GET_TEMP_MEM,          | Allocates listed dynamic storage
91 0223 1 |     DBG$NMATCH,                | Match input string
92 0224 1 |     DBG$NNEXT_WORD,            | Get next word in input
93 0225 1 |     DBG$OUT_MESSAGE : NOVALUE, | Called as an action routine by SYS$PUTMSG to
94 0226 1 |                                | output the error message.
95 0227 1 |     SYS$PUTMSG : ADDRESSING_MODE (GENERAL); | System message output routine
96 0228 1 |
97 0229 1 | EXTERNAL
98 0230 1 |     DBG$GL_DEVELOPER: BITVECTOR[]; | Developer flags
99 0231 1 |
100 0232 1 |
101 0233 1 | LITERALS
102 0234 1 |
103 0235 1 | Used for communication between PARSE_DUMP and EXECUTE_DUMP.
104 0236 1 |
105 0237 1 | LITERAL
106 0238 1 |     DUMP_MIN = 0,
107 0239 1 |     DUMP_HASH = 0,
108 0240 1 |     DUMP_SAT = 1,
109 0241 1 |     DUMP_GST = 2,
110 0242 1 |     DUMP_MAX = 2;
```

```

112 0243 1 GLOBAL ROUTINE DBG$NPARSE_DUMP (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
113 0244 1
114 0245 1 FUNCTION
115 0246 1     This routine parses the DUMP command. This command dumps internal
116 0247 1     DEBUG data structures. The command is only available to developers.
117 0248 1
118 0249 1 INPUTS
119 0250 1     INPUT_DESC      - The remaining command string.
120 0251 1     VERB_NODE       - Pointer to partially constructed parse tree
121 0252 1     MESSAGE_VECT   - Error message vector
122 0253 1
123 0254 1 OUTPUTS
124 0255 1     Information is printed at the terminal.
125 0256 1     The input string is updated to point beyond what we picked up.
126 0257 1     A return status is returned.
127 0258 1
128 0259 1 BEGIN
129 0260 1 MAP
130 0261 1     INPUT_DESC: REF DBG$STG_DESC;
131 0262 1     VERB_NODE: REF DBG$VERB_NODE;
132 0263 1
133 0264 1 BIND
134 0265 1     DBG$CS_CR      = UPLIT BYTE (1, DBG$K_CAR_RETURN),
135 0266 1     DBG$CS_GST      = UPLIT BYTE (3, 'GST'),
136 0267 1     DBG$CS_HASH     = UPLIT BYTE (4, 'HASH'),
137 0268 1     DBG$CS_SAT      = UPLIT BYTE (3, 'SAT');
138 0269 1
139 0270 1 ! Check developer flag 0. This enables the DUMP command.
140 0271 1
141 0272 1 IF NOT .DBG$GL_DEVELOPER[0]
142 0273 1 THEN
143 0274 1     BEGIN
144 0275 1     .MESSAGE_VECT = DBG$NSYNTAX_ERROR(UPLIT BYTE(%ASCIC 'DUMP'));
145 0276 1     RETURN ST$K_SEVERE;
146 0277 1     END;
147 0278 1
148 0279 1 ! Pick up the keyword. At the moment, we only support DUMP HASH,
149 0280 1 ! but more keywords may be added later.
150 0281 1
151 0282 1 SELECTONE TRUE OF
152 0283 1 SET
153 0284 1     ! DUMP GST
154 0285 1     !
155 0286 1     [DBG$NMATCH (.INPUT_DESC, DBG$CS_GST, 1)]:
156 0287 1     BEGIN
157 0288 1     VERB_NODE[DBG$L_VERB_OBJECT_PTR] = DUMP_GST;
158 0289 1     END;
159 0290 1
160 0291 1     ! DUMP HASH.
161 0292 1     !
162 0293 1     [DBG$NMATCH (.INPUT_DESC, DBG$CS_HASH, 1)]:
163 0294 1     BEGIN
164 0295 1     VERB_NODE[DBG$L_VERB_OBJECT_PTR] = DUMP_HASH;
165 0296 1     END;
166 0297 1
167 0298 1     ! DUMP SAT.
168 0299 1

```



```
169 0300 2
170 0301 2
171 0302 2
172 0303 2
173 0304 2
174 0305 2
175 0306 2
176 0307 2
177 0308 2
178 0309 2
179 0310 4
180 0311 4
181 0312 4
182 0313 4
183 0314 4
184 0315 4
185 0316 3
186 0317 2
187 0318 2
188 0319 2
189 0320 1

!
[DBG$NMATCH (.INPUT_DESC, DBG$CS_SAT, 1)]:
  BEGIN
  VERB_NODE[DBG$L_VERB_OBJECT_PTR] = DUMP_SAT;
  END;

! Any other DUMP argument is a syntax error.
[OTHERWISE]:
  BEGIN
  .MESSAGE VECT = (
    IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
    THEN
      DBG$NMAKE_ARG_VECT(DBG$NEEDMORE)
    ELSE
      DBG$NSYNTAX_ERROR(DBG$NNEXT_WORD(.INPUT_DESC));
    RETURN ST$K_SEVERE;
  END;
  TES;
  RETURN ST$K_SUCCESS;
END;
```

```
.TITLE DBGNERMSG
.IDENT \V04-000\

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0

      OD 01 00000 P.AAA: .BYTE 1, 13
      54 53 03 00002 P.AAB: .BYTE 3
      48 53 47 00003 .ASCII \GST\
      50 4D 04 00006 P.AAC: .BYTE 4
      54 41 48 00007 .ASCII \HASH\
      54 41 03 0000B P.AAD: .BYTE 3
      50 4D 54 41 53 0000C .ASCII \SAT\
      55 44 04 0000F P.AAE: .ASCII <4>\DUMP\

      DBG$CS_CR= P.AAA
      DBG$CS_GST= P.AAB
      DBG$CS_HASH= P.AAC
      DBG$CS_SAT= P.AAD

      .EXTRN DBG$ANALYZE_HASH
      .EXTRN DBG$DUMP_GLOBAL
      .EXTRN DBG$DUMP_SAT, DBG$GET_TEMPMEM
      .EXTRN DBG$NMATCH, DBG$NNEXT_WORD
      .EXTRN DBG$OUT_MESSAGE
      .EXTRN SY$PUTMSG, DBG$GL_DEVELOPER

.PSECT DBG$CODE,NOWRT, SHR, PIC,0

      54 00000000G 00 001C 00000 .ENTRY DBG$NPARSE_DUMP, Save R2,R3,R4
      53 00000000G EF 9E 00002 MOVAB DBG$NMATCH, R4
      04 00000000G 00 E8 00010 MOVAB P.AAE, R3
      53 DD 00017 BLBS DBG$GL_DEVELOPER, 1$
      73 11 00019 PUSHL R3
      01 DD 0001B 1$: BRB 6$
      PUSHL #1
```

.....

: 0243
: 0272
: 0275
: 0287

		F3	A3	9F	0001D	PUSHAB	DBG\$CS_GST		
	52	04	AC	D0	00020	MOVL	INPUT_DESC, R2		
			52	DD	00024	PUSHL	R2		
	64		03	FB	00026	CALLS	#3, DBG\$NMATCH		
	01		50	D1	00029	CMPL	R0, #1		
			0A	12	0002C	BNEQ	2\$		
	50	08	AC	D0	0002E	MOVL	VERB_NODE, R0		0289
08	A0		02	D0	00032	MOVL	#2, 8(R0)		
			63	11	00036	BRB	8\$		0282
			01	DD	00038	PUSHL	#1		0294
		F7	A3	9F	0003A	PUSHAB	DBG\$CS_HASH		
			52	DD	0003D	PUSHL	R2		
	64		03	FB	0003F	CALLS	#3, DBG\$NMATCH		
	01		50	D1	00042	CMPL	R0, #1		
			09	12	00045	BNEQ	3\$		
	50	08	AC	D0	00047	MOVL	VERB_NODE, R0		0296
		08	A0	D4	0004B	CLRL	8(R0)		
			4B	11	0004E	BRB	8\$		0282
			01	DD	00050	PUSHL	#1		0301
		FC	A3	9F	00052	PUSHAB	DBG\$CS_SAT		
			52	DD	00055	PUSHL	R2		
	64		03	FB	00057	CALLS	#3, DBG\$NMATCH		
	01		50	D1	0005A	CMPL	R0, #1		
			0A	12	0005D	BNEQ	4\$		
	50	08	AC	D0	0005F	MOVL	VERB_NODE, R0		0303
08	A0		01	D0	00063	MOVL	#1, 8(R0)		
			32	11	00067	BRB	8\$		0282
			01	DD	00069	PUSHL	#1		0311
		F1	A3	9F	0006B	PUSHAB	DBG\$CS_CR		
			52	DD	0006E	PUSHL	R2		
	64		03	FB	00070	CALLS	#3, DBG\$NMATCH		
	0D		50	E9	00073	BLBC	R0, 5\$		
		000280D0	8F	DD	00076	PUSHL	#164048		0313
0000V	CF		01	FB	0007C	CALLS	#1, DBG\$NMAKE_ARG_VECT		
			10	11	00081	BRB	7\$		
			52	DD	00083	PUSHL	R2		0315
00000000G	00		01	FB	00085	CALLS	#1, DBG\$NNEXT_WORD		
			50	DD	0008C	PUSHL	R0		
0000V	CF		01	FB	0008E	CALLS	#1, DBG\$NSYNTAX_ERROR		
0C	BC		50	D0	00093	MOVL	R0, @MESSAGE_VECT		0310
	50		04	D0	00097	MOVL	#4, R0		0316
			04	0009A	RET				
	50		01	D0	0009B	MOVL	#1, R0		0319
			04	0009E	RET				0320

; Routine Size: 159 bytes, Routine Base: DBG\$CODE + 0000


```
191 0321 1 GLOBAL ROUTINE DBG$NEXECUTE_DUMP (VERB_NODE, MESSAGE_VECT) =
192 0322 1
193 0323 1 FUNCTION
194 0324 1     Performs the action associated with the DUMP command.
195 0325 1
196 0326 1 INPUTS
197 0327 1     VERB_NODE      - A pointer to the command tree
198 0328 1     MESSAGE_VECT   - Error message vector
199 0329 1
200 0330 1 OUTPUTS
201 0331 1     Information about internal DEBUG data structures will be printed
202 0332 1     at the terminal. A status code is returned.
203 0333 1
204 0334 2 BEGIN
205 0335 2 MAP
206 0336 2     VERB_NODE: REF DBG$VERB_NODE;
207 0337 2
208 0338 2     ! Case on the DUMP keyword. DUMP HASH is the only one we currently
209 0339 2     support.
210 0340 2
211 0341 2 CASE .VERB_NODE[DBG$L_VERB_OBJECT_PTR] FROM DUMP_MIN TO DUMP_MAX OF
212 0342 2     SET
213 0343 2
214 0344 2     [DUMP_GST]:
215 0345 2         DBG$DUMP_GLOBAL();
216 0346 2
217 0347 2     [DUMP_HASH]:
218 0348 2         DBG$ANALYZE_HASH();
219 0349 2
220 0350 2     [DUMP_SAT]:
221 0351 2         DBG$DUMP_SAT();
222 0352 2
223 0353 2     [INRANGE, OUTRANGE]:
224 0354 2         $DBG_ERROR('DBGNERMSG\DBG$NEXECUTE_DUMP');
225 0355 2
226 0356 2     TES:
227 0357 2     RETURN ST$K_SUCCESS;
228 0358 1 END;
```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

24 47 42 44 5C 47 53 4D 52 45 4E 47 42 44 1B 00014 P.AAF: .ASCII <27>\DBGNERMSG\<92>\DBG\$NEXECUTE_DUMP\ :
50 4D 55 44 5F 45 54 55 43 45 58 45 4E 00023 :

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

```
0000 00000
50 04 AC DO 00002 .ENTRY DBG$NEXECUTE_DUMP, Save nothing : 0321
00 08 AO CF 00006 MOVL VERB_NODE, R0 : 0341
001D 002F 0026 0000B 1$: CASEL 8(R0, #0, #2
WORD 3$-1$,-
4$-1$,-
2$-1$
PUSHAB P.AAF : 0354
```


DBGNERMSG
V04-000

B 7
16-Sep-1984 01:42:49
14-Sep-1984 12:17:11

VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGNERMSG.B32;1

Page 8
(4)

		01	DD	00017		PUSHL	#1	
		8F	DD	00019		PUSHL	#164706	
00000000G	00	03	FB	0001F		CALLS	#3, LIB\$SIGNAL	
		19	11	00026		BRB	5\$	
00000000G	00	00	FB	00028	2\$:	CALLS	#0, DBG\$DUMP_GLOBAL	0345
		10	11	0002F		BRB	5\$	
00000000G	00	00	FB	00031	3\$:	CALLS	#0, DBG\$ANALYZE_HASH	0348
		07	11	00038		BRB	5\$	
00000000G	00	00	FB	0003A	4\$:	CALLS	#0, DBG\$DUMP_SAT	0351
	50	01	D0	00041	5\$:	MOVL	#1, R0	0357
		04	00	00044		RET		0358

; Routine Size: 69 bytes, Routine Base: DBG\$CODE + 009F


```
230 0359 1 GLOBAL ROUTINE DBG$NOUT_INFO (ERROR_CODE) =
231 0360 1
232 0361 1 ++
233 0362 1 FUNCTIONAL DESCRIPTION:
234 0363 1
235 0364 1 This routine outputs an informational message to the user's terminal and/or
236 0365 1 log file.
237 0366 1
238 0367 1 This routine will not output message that do not have an informational
239 0368 1 level of severity.
240 0369 1
241 0370 1 FORMAL PARAMETERS:
242 0371 1
243 0372 1 error_code - A longword containing an integer value corresponding
244 0373 1 to a DEBUG info message code
245 0374 1
246 0375 1 [fao_count] - A longword containing the number of fao arguments supplied
247 0376 1 in conjunction with the first message code. This optional
248 0377 1 parameter MUST be supplied if ANY fao arguments are supplied.
249 0378 1
250 0379 1 [fao_first, ...] - A longword containing an fao argument to be incorporated
251 0380 1 into the info message text
252 0381 1
253 0382 1 [next_code, next_count, next_fao, ...]
254 0383 1
255 0384 1 - The next message code, fao_count, fao_argument sequence.
256 0385 1
257 0386 1 IMPLICIT INPUTS:
258 0387 1
259 0388 1 NONE
260 0389 1
261 0390 1 IMPLICIT OUTPUTS:
262 0391 1
263 0392 1 NONE
264 0393 1
265 0394 1 ROUTINE VALUE:
266 0395 1
267 0396 1 An unsigned integer longword completion code
268 0397 1
269 0398 1 COMPLETION CODES:
270 0399 1
271 0400 1 sts$success (1) - Success. Informational message output.
272 0401 1
273 0402 1 sts$severe (4) - Failure. Message not an info and not output.
274 0403 1
275 0404 1 SIDE EFFECTS:
276 0405 1
277 0406 1 Outputs an informational message(s) to the user's terminal and/or log file.
278 0407 1
279 0408 1 --
280 0409 2 BEGIN
281 0410 2
282 0411 2 BUILTIN
283 0412 2 ACTUALCOUNT,
284 0413 2 ACTUALPARAMETER;
285 0414 2
286 0415 2 LOCAL
```



```
287 0416 2 NUM_ACTUALS, ! Number of actual parameters
288 0417 2 ! Loop counter
289 0418 2 ERROR_VECT, ! Message vector
290 0419 2 ARG_VECT : REF VECTOR; ! The message argument vector
291 0420 2
292 0421 2
293 0422 2 ! Make sure that the message code corresponds to an info
294 0423 2
295 0424 2 IF .error_code <0, 3, 0> NEQ sts$k_info
296 0425 2 THEN
297 0426 2 RETURN sts$k_severe;
298 0427 2
299 0428 2
300 0429 2 ! Make the argument vector
301 0430 2
302 0431 2 num_actuals = actualcount ();
303 0432 2
304 0433 2 arg_vect = dbg$get_tempmem(.num_actuals + 1);
305 0434 2 arg_vect [0] = .num_actuals;
306 0435 2
307 0436 2 INCR i FROM 1 TO .num_actuals
308 0437 2 DO
309 0438 2 arg_vect [.i] = actualparameter (.i);
310 0439 2
311 0440 2
312 0441 2 ! Output the message
313 0442 2
314 0443 2 dbg$nout_arg_vect (.arg_vect);
315 0444 2
316 0445 2 RETURN sts$k_success;
317 0446 2
318 0447 1 END; ! End of dbg$nout_info
```

03	04	AC	03	0004 00000	.ENTRY	DBG\$NOUT_INFO, Save R2	: 0359
			50	00 ED 00002	CMPZV	#0, #3, ERROR_CODE, #3	: 0424
			52	04 13 00008	BEQL	1\$: 0426
			60	04 D0 0000A	MOVL	#4, R0	: 0431
			6041	04 0000D	RET		: 0433
			51	6C 9A 0000E 1\$:	MOVZBL	(AP), NUM_ACTUALS	: 0434
			52	A2 9F 00011	PUSHAB	1(NUM_ACTUALS)	: 0436
			50	01 FB 00014	CALLS	#1, DBG\$GET_TEMPMEM	: 0438
			51	52 D0 0001B	MOVL	NUM_ACTUALS, (ARG_VECT)	: 0443
			50	51 D4 0001E	CLRL	I	: 0445
			50	05 11 00020	BRB	3\$: 0447
			51	6C41 D0 00022 2\$:	MOVL	(AP)[I], (ARG_VECT)[I]	
			50	52 F3 00027 3\$:	AOBLEQ	NUM_ACTUALS, I, 2\$	
			50	DD 0002B	PUSHL	ARG_VECT	
			50	01 FB 0002D	CALLS	#1, DBG\$NOUT_ARG_VECT	
			50	01 D0 00032	MOVL	#1, R0	
			50	04 00035	RET		

; Routine Size: 54 bytes, Routine Base: DBG\$CODE + 00E4

DBGNERMSG
V04-000

E 7
16-Sep-1984 01:42:49
14-Sep-1984 12:17:11

VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGNERMSG.B32;1

Page 11
(5)

; 319

0448 1

```
321 0449 1 GLOBAL ROUTINE DBG$NMAKE_ARG_VECT (ERROR_CODE) =
322 0450 1
323 0451 1 ++
324 0452 1 FUNCTIONAL DESCRIPTION:
325 0453 1
326 0454 1     Creates a message argument vector as described on page 4-119 of
327 0455 1     the VAX/VMS system reference, volume 1A.
328 0456 1
329 0457 1     This routine ALWAYS returns the address of a message argument vector.
330 0458 1
331 0459 1 FORMAL PARAMETERS:
332 0460 1
333 0461 1     error_code      - A longword containing an integer corresponding to a
334 0462 1                     DEBUG message code
335 0463 1
336 0464 1     [fao_count]      - A longword containing the number of fao arguments supplied
337 0465 1                     in conjunction with error_code. This optional parameter
338 0466 1                     MUST be supplied if ANY fao arguments are supplied.
339 0467 1
340 0468 1     [fao_first, ...] - A longword containing an FAO argument to be inserted
341 0469 1                     into the text of a DEBUG message
342 0470 1
343 0471 1     Note that the above sequence may be repeated to construct an argument
344 0472 1     vector for concatenated messages.
345 0473 1
346 0474 1 IMPLICIT INPUTS:
347 0475 1
348 0476 1     NONE
349 0477 1
350 0478 1 IMPLICIT OUTPUTS:
351 0479 1
352 0480 1     NONE
353 0481 1
354 0482 1 ROUTINE VALUE:
355 0483 1
356 0484 1     An unsigned integer longword corresponding to the address of a message
357 0485 1     argument vector.
358 0486 1
359 0487 1 COMPLETION CODES:
360 0488 1
361 0489 1     NONE
362 0490 1
363 0491 1 SIDE EFFECTS:
364 0492 1
365 0493 1     NONE
366 0494 1
367 0495 1 --
368 0496 2 BEGIN
369 0497 2
370 0498 2 BUILTIN
371 0499 2     ACTUALCOUNT,
372 0500 2     ACTUALPARAMETER;
373 0501 2
374 0502 2 LOCAL
375 0503 2     NUM_ACTUALS,           ! Number of actual parameters
376 0504 2     I,                    ! Loop counter
377 0505 2     ERROR_VECT,         ! Error vector pointer
```



```
: 378      0506 2      ARG_VECT : REF VECTOR;      ! Messagr argument vector
: 379      0507 2
: 380      0508 2
: 381      0509 2      ! Make the argument vector
: 382      0510 2      !
: 383      0511 2      num_actuals = actualcount ();
: 384      0512 2
: 385      0513 2      arg_vect = dbg$get_tempmem(.num_actuals + 1);
: 386      0514 2      arg_vect [0] = .num_actuals;
: 387      0515 2
: 388      0516 2      INCR i FROM 1 TO .num_actuals
: 389      0517 2      DO
: 390      0518 2          arg_vect [.i] = actualparameter (.i);
: 391      0519 2
: 392      0520 2      RETURN .arg_vect;
: 393      0521 2
: 394      0522 1      END;      ! End of dbg$make_arg_vect
```

			0004 00000	.ENTRY	DBG\$MAKE_ARG_VECT, Save R2	: 0449
	52		6C 9A 00002	MOVZBL	(AP), NUM_ACTUALS	: 0511
		01	A2 9F 00005	PUSHAB	1(NUM_ACTUALS)	: 0513
00000000G	00		01 FB 00008	CALLS	#1, DBG\$GET_TEMPMEM	:
	60		52 D0 0000F	MOVL	NUM_ACTUALS, (ARG_VECT)	: 0514
			51 D4 00012	CLRL	I	: 0516
			05 11 00014	BRB	2\$:
	6041	6C41	D0 00016 1\$:	MOVL	(AP)[I], (ARG_VECT)[I]	: 0518
F7	51	52	F3 0001B 2\$:	AOBLEQ	NUM_ACTUALS, I, 1\$:
			04 0001F	RET		: 0522

; Routine Size: 32 bytes, Routine Base: DBG\$CODE + 011A

; 395 0523 1

```
397 0524 1 GLOBAL ROUTINE DBG$NOUT_ARG_VECT (ARGUMENT_VECT) : NOVALUE =
398 0525 1
399 0526 1 ++
400 0527 1 FUNCTIONAL DESCRIPTION:
401 0528 1
402 0529 1     Outputs the DEBUG error message corresponding to the input message
403 0530 1     argument vector to the user's terminal and/or log file.
404 0531 1
405 0532 1     This routine should be invoked directly only by the DEBUG CLI.
406 0533 1
407 0534 1 FORMAL PARAMETERS:
408 0535 1
409 0536 1     argument_vect  - A longword containing the address of a message argument
410 0537 1                    vector as described on page 4-119 of the VAX/VMS system
411 0538 1                    reference, volume 1A
412 0539 1
413 0540 1 IMPLICIT INPUTS:
414 0541 1
415 0542 1     The parameter argument_vect is set to 0 after the output
416 0543 1
417 0544 1 IMPLICIT OUTPUTS:
418 0545 1
419 0546 1     NONE
420 0547 1
421 0548 1 ROUTINE VALUE:
422 0549 1
423 0550 1     NONE
424 0551 1
425 0552 1 COMPLETION CODES:
426 0553 1
427 0554 1     NONE
428 0555 1
429 0556 1 SIDE EFFECTS:
430 0557 1
431 0558 1     Writes a DEBUG error message to the user's terminal and/or log file.
432 0559 1
433 0560 1     This routine signals a debugbug if there is no message to output.
434 0561 1
435 0562 1 --
436 0563 2 BEGIN
437 0564 2
438 0565 2     ! Check for no error message to output.
439 0566 2     !
440 0567 2     IF .argument_vect EQLA 0
441 0568 2     THEN
442 0569 2         $DBG_ERROR('DBGNERMSG\DBG$NOUT_ARG_VECT');
443 0570 2
444 0571 2     ! Output the message.
445 0572 2     !
446 0573 2     SYSS$PUTMSG (.argument_vect, dbg$out_message, 0);
447 0574 2
448 0575 2 RETURN;
449 0576 2
450 0577 2
451 0578 1 END;          ! End of dbg$nout_arg_vect
```



```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
24 47 42 44 5C 47 53 4D 52 45 4E 47 42 44 1B 00030 P.AAG: .ASCII <27>\DBGNERMSG\<92>\DBG$NOUT_ARG_VECT\
54 43 45 56 5F 47 52 41 5F 54 55 4F 4E 0003F
;

.PSECT DBG$CODE,NOWRT, SHR, PIC,0
;ENTRY DBG$NOUT_ARG_VECT, Save nothing ; 0524
;ISTL ARGUMENT_VECT ; 0568
;BNEQ 1$ ;
;PUSHAB P.AAG ; 0570
;PUSHL #1 ;
;PUSHL #164706 ;
;CALLS #3, LIB$SIGNAL ;
;CLRL -(SP) ; 0574
;PUSHAB DBG$OUT_MESSAGE ;
;PUSHL ARGUMENT_VECT ;
;CALLS #3, SYSS$PUTMSG ;
;RET ; 0578

0000 00000
04 AC D5 00002
15 12 00005
00000000' EF 9F 00007
01 DD 0000D
00028362 8F DD 0000F
00000000G 00 03 FB 00015
7E D4 0001C 1$:
00000000G 00 00 9F 0001E
04 AC DD 00024
00000000G 00 03 FB 00027
04 04 0002E

```

; Routine Size: 47 bytes, Routine Base: DBG\$CODE + 013A

; 452 0579 1

```

: 454      0580 1 GLOBAL ROUTINE DBG$NSYNTAX_ERROR (WORD_STRING) =
: 455      0581 1
: 456      0582 1 ++
: 457      0583 1 FUNCTIONAL DESCRIPTION:
: 458      0584 1
: 459      0585 1     Called as a result of the detection of a syntax error. Constructs a
: 460      0586 1     syntax error message argument vector.
: 461      0587 1
: 462      0588 1 FORMAL PARAMETERS:
: 463      0589 1
: 464      0590 1     word_string -           The word corresponding to the syntax error
: 465      0591 1
: 466      0592 1 IMPLICIT INPUTS:
: 467      0593 1
: 468      0594 1     NONE
: 469      0595 1
: 470      0596 1 IMPLICIT OUTPUTS:
: 471      0597 1
: 472      0598 1     The message argument vector associated with the syntax error. This includes
: 473      0599 1     an ascii string descriptor which points to the syntax error string.
: 474      0600 1
: 475      0601 1 ROUTINE VALUE:
: 476      0602 1
: 477      0603 1     The beginning address of the message argument vector
: 478      0604 1
: 479      0605 1 COMPLETION CODES:
: 480      0606 1
: 481      0607 1     NONE
: 482      0608 1
: 483      0609 1 SIDE EFFECTS:
: 484      0610 1
: 485      0611 1     NONE
: 486      0612 1
: 487      0613 1 --
: 488      0614 2 BEGIN
: 489      0615 2
: 490      0616 2 MAP
: 491      0617 2     WORD_STRING : REF VECTOR [,BYTE];
: 492      0618 2
: 493      0619 2 LOCAL
: 494      0620 2     ERROR_VECT,           ! Error message pointer
: 495      0621 2     STRING_DESC : REF dbg$stg_desc; ! String descriptor for error message
: 496      0622 2
: 497      0623 2
: 498      0624 2     ! Get storage for the string descriptor
: 499      0625 2     !
: 500      0626 2     string_desc = dbg$get_tempmem(2);
: 501      0627 2
: 502      0628 2
: 503      0629 2     ! make the string descriptor
: 504      0630 2     !
: 505      0631 2     string_desc [dsc$w_length] = .word_string [0];
: 506      0632 2     string_desc [dsc$a_pointer] = word_string [1];
: 507      0633 2
: 508      0634 2
: 509      0635 2     ! Construct the vector and return it.
: 510      0636 2
: 511      0636 2
```



```

: 511      0637 2      error_vect = dbg$make_arg_vect (dbg$_syntax, 1, .string_desc);
: 512      0638 2
: 513      0639 2      RETURN .error_vect;
: 514      0640 2
: 515      0641 1      END;          ! End of dbg$_syntax_error

```

```

                                0000 00000
                                02 DD 00002
                                01 FB 00004
00000000G 00                  04 BC 9B 0000B
04 A0      04 AC              01 C1 0000F
                                50 DD 00015
                                01 DD 00017
                                8F DD 00019
                                03 FB 0001F
                                04 00023
                                8E AF 00028238

```

```

.ENTRY  DBG$NSYNTAX_ERROR, Save nothing
PUSHL  #2
CALLS  #1, DBG$GET_TEMPMEM
MOVZBW @WORD STRING, (STRING_DESC)
ADDL3  #1, WORD STRING, 4(STRING_DESC)
PUSHL  STRING_DESC
PUSHL  #1
PUSHL  #164408
CALLS  #3, DBG$NMAKE_ARG_VECT
RET

```

```

: 0580
: 0626
: 0631
: 0632
: 0637
:
: 0641

```

; Routine Size: 36 bytes, Routine Base: DBG\$CODE + 0169

```

: 516      0642 1

```

: 518 0643 1 END
: 519 0644 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
DBG\$PLIT	76	NOVEC,NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
DBG\$CODE	397	NOVEC,NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	5	0	1000	00:01.9
_\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
_\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	12	0	97	00:02.0
_\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	0	0	31	00:00.4
_\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	3	0	22	00:00.3

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGNERMSG/OBJ=OBJ\$:DBGNERMSG MSRC\$:DBGNERMSG/UPDATE=(ENH\$:DBGNERMSG)

: Size: 397 code + 76 data bytes
: Run Time: 00:13.5
: Elapsed Time: 01:04.6
: Lines/CPU Min: 2872
: Lexemes/CPU-Min: 7378
: Memory Used: 87 pages
: Compilation Complete

0087 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGNMSG
LIS

DBGNHELP
LIS

DBGNPARSE
LIS

DBGNEXCTE
LIS

DBGNPNP
LIS